Teoretyczne podstawy informatyki

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. letni 2025/26

inf.ug.edu.pl/~amb/TPI

Dalsze operacje na językach bezkontekstowych

Podstawienie

- dany alfabet A i język $L \subseteq A^*$
- dany być może inny alfabet Σ i dla każdej litery $a \in A$ język $L_a \subseteq \Sigma^*$
- dla słowa $u \in A^*$ definiujemy język $L_u = L_{u_1} \cdot \ldots \cdot L_{u_\ell}$
- dla całego języka L definiujemy $subs(L) = \bigcup_{u \in L} L_u$
- tw. jeśli L i wszystkie języki L_a są bezkontekstowe, to $subs(L)$ jest językiem bezkontekstowym nad Σ
- dw.: możemy założyć, że zmienne gramatyk dla L i L_a są rozłączne
  - symbolem początkowym będzie symbol początkowy dla L
  - jeśli w produkcji gramatyki dla L występuje symbol terminalny $a \in A$, zastępujemy do symbolem początkowym S_a języka L_a
  - symbolami końcowymi będzie zbiór Σ
  - konstrukcja analogiczna do języków regularnych – wystąpienie a w słowie nie kończy produkcji, jest kontynuowane w kolejnym języku

Podstawienie, przykład

- dane języki bezkontekstowe L_a i L_b
  rozważmy języki nad $\{a, b\}$: $L_{sum} = \{a, b\}$, $L_{konk} = \{ab\}$, $L_{iter} = \{a^n \mid n \in \mathbb{N}\}$
- podstawienie $a \mapsto L_a$, $b \mapsto L_b$ zastosowane do powyższych języków prowadzi do $L_a \cup L_b$, $L_a L_b$, L_a^*
- czyli dowód, że klasa języków bezkontekstowych jest zamknięta na sumę, konkatenację i iterację jest wnioskiem z twierdzenia o podstawieniu

Homomorfizm

- homomorfizm $h : A \to \Sigma^*$ to szczególne podstawienie $L_a = \{h(a)\}$ będziemy automatycznie przedłużać na słowa, $h : A^* \to \Sigma^*$, $h(u) = h(u_1) \dots h(u_\ell)$
- tw.: obraz i przeciwobraz języka bezkontekstowego jest językiem bezkontekstowym
  - dw.: obraz: było przed chwilą
  - przeciwobraz: dowód jest znacznie bardziej złożony niż dla języków regularnych, idea jest podobna
    w automacie ze stosem rozpoznającym język nad Σ definiujemy przejścia etykietowane symbolami z A analizując ruchy automatu przy wejściu $h(a)$
    konieczne jest zadbanie o prawidłową obsługę stosu, co jest możliwe, szczegóły pomijamy

Podstawienia i homomorfizmy dla języków klasy DPDA

- tw.: przeciwobraz języka rozpoznawanego przez DPDA należy do tej samej klasy
  - dw.: w dowodzie budowany jest nowy automat, jeśli oryginalny był deterministyczny, to nowy też będzie
- tw.: obraz przy homomorfiźmie języka rozpoznawanego przez DPDA nie należy do tej klasy, podstawienie również
  - dw.: załóżmy, że dane są dwa języki L_i nad alfabetem $\{a, b\}$, suma $1L_1 \cup 2L_2$ jest językiem nad $\{1, 2, a, b\}$ rozpoznawanym przez DPDA; homomorfizm $1 \mapsto 2$ skleja pierwsze litery i obraz $2L_1 \cup 2L_2$ nie musi być rozpoznawany
  - ten sam pomysł L_1 i $L2$ pokazuje, że konkatenacja z 1^* nie jest klasy DPDA

Algorytmy decyzyjne

Problemy dotyczące języków

- dane języki L, L_1, $L_2 \subseteq A^*$, słowo $w \in A^*$, możemy postawić pytania
  - czy język $L \neq \emptyset$, jest niepusty?
  - czy język $|L| < \infty$, jest skończony?
  - czy $L_1 = L_2$, języki są równe?
  - ogólniej, czy $L_1 \subseteq L_2$?
  - czy $w \in L$, słowo należy do języka?
- oprócz samej odpowiedzi interesuje nas też algorytm o jak najmniejszej złożoności
- odpowiedzi zależą od
  - klasy języków: regularne, bezkontekstowe, inne klasy
  - sposobu zdefiniowania języka: automat deterministyczny, niedeterministyczny, gramatyka, w jednej z postaci normalnych

Problem pustości języka $L = \emptyset$?

- język regularny dany jest automatem deterministycznym lub nie, zaakceptowane słowo wyznacza ścieżkę od stanu początkowego do akceptującego
  - czyli wystarczy wykonać algorytm szukania ścieżki w grafie, np. wgłąb, i sprawdzić, czy znajdziemy stan akceptujący na końcu ścieżki, niedeterminizm automatu w niczym nie przeszkadza
  - dla wyrażenia regularnego można łatwo zbudować automat
  - alternatywnie gwarancję niepustości daje ε lub dowolna litera alfabetu w wyrażeniu i można przeanalizować całość
- język bezkontekstowy dany jest gramatyką
  - algorytm usuwania zbędnych symboli nieterminalnych albo pokaże, że nie ma żadnych produkcji albo udowodni, że język jest niepusty
  - język zdefiniowany automatem ze stosem wymaga zbudowania gramatyki, koszt może być nietrywialny ale jest wielomianowy

Problem skończoności języka $|L| < \infty$?

- język regularny dany jest automatem, zaakceptowane słowo wyznacza ścieżkę od stanu początkowego do akceptującego
  - język ma nieskończenie wiele słów w.t.w. na jakiejś ścieżce istnieje cykl (lemat o pompowaniu)
  - wyrażenie regularne można przekształcić w automat
  - można też analizować wyrażenie, źródłem nieskończoności jest operator Kleene'go ale stosowany do podwyrażenia niepustego
- język bezkontekstowy zdefiniowany gramatyką Chomsky'ego
  - produkcja $A \to BC$ buduje krawędzie $A \to B$, $A \to C$ w grafie symboli nieterminalnych
  - tw: język jest nieskończony w.t.w. istnieje cykl osiągalny z symbolu początkowego
  - dw.: dość oczywisty

Problem przynależności słowa $w \in L$?

- jeśli język regularny dany jest automatem deterministycznym, wystarczy wykonać automat na słowie w
  - jeśli automat definiujący jest niedeterministyczny, po odczytaniu każdej litery słowa trzeba rozpatrywać możliwe stany automatu, złożoność mniejsza niż determinizacja całości
  - jeśli język definiowany jest wyrażeniem regularnym, to daje ono automat niedeterministyczny i dalej j.w.
- dla języka definiowanego gramatyką bezkontekstową można stosować naiwny algorytm, w którym wykonujemy obliczenia automatu ze stosem testując wszystkie możliwe produkcje przy odczycie każdej litery
  - zdecydowanie lepszą złożoność $O(n^3)$ ma algorytm CYK (Cock-Younger-Kasami)
  - niektóre gramatyki gwarantują mniejszą złożoność tego algorytmu

Algorytm CYK

- problem: dana gramatyka bezkontekstowa w postaci Chomsky'ego (tzn. produkcje są postaci $A \to BC$ lub $A \to a$), sprawdzić czy słowo $w = w_1 \dots w_n$ jest generowane przez tę gramatykę
- budujemy dwuwymiarową macierz CYK
  - w komórce $CYK[i,j]$ są zmienne gramatyki generujące podsłowo długości i zaczynające się w miejscu j, $w_j \dots w_{j+i-1}$
  - dla $i = 1$ mogą być użyte jedynie produkcje $A \to a$
  - gdy słowa długości $< m$ zostały rozpatrzone, sprawdzamy, czy $B \in CYK[k,j]$, $C \in CYK[m-k, j+k]$ oraz $A \to BC$ jeśli tak, to $A \in CYK[m,j]$ – generuje słowo długości m w miejscu j
  - $w \in L$ w.t.w. $S \in CYK[n,1]$
- jest $O(n^2)$ komórek w macierzy, każdą wypełniamy kosztem $O(n)$
- złożoność $O(n^3)$

Algorytm CYK, przykład $(2 + 3) * 5$

- gramatyka wyrażeń arytmetycznych w postaci Chomsky'ego

$W \to 0 \mid \ldots \mid 9 \mid (P \mid SR \mid WT$	wyrażenie
$P \to W)$	reszta nawiasu
$R \to *C$	reszta mnożenia
$T \to +S$	reszta dodawania
$S \to 0 \mid \ldots \mid 9 \mid (P \mid SR$	składnik
$C \to 0 \mid \ldots \mid 9 \mid (P$	czynnik

- macierz CYK

	1	2	3	4	5	6	7
1	(	WSC	$+$	WSC	)	$*$	WSC
2			T	P		R	
3		W					
4		P					
5	WSC						
6							
7	WS						